RL-TR-95-120
Final Technical Report
July 1995

INNOVATIVE APPROACH TO DEVELOPMENT OF A VARIABLE CONFIGURATION MULTISENSOR TEST CAPABILITY

CSC Professional Services Group

Gerald A. Bright, Robert S. Mandry, and Mark D. Barnell

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

19951012 099

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-95-120 has been reviewed and is approved for publication.

APPROVED: Richard R. Gassner

RICHARD R. GASSNER
Project Engineer

FOR THE COMMANDER: Donald W. Hanson

DONALD W. HANSON
Director of Surveillance & Photonics

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (OCSM) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1995	Mar 94 - Oct 94

4. TITLE AND SUBTITLE

INNOVATIVE APPROACH TO DEVELOPMENT OF A VARIABLE CONFIGURATION MULTISENSOR TEST CAPABILITY

5. FUNDING NUMBERS

C - F30602-94-C-0058
PE - 61102F
PR - 2304
TA - E8
WU - PB

6. AUTHOR(S)

Gerald A. Bright, Robert S. Mandry, and Mark D. Barnell

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

CSC Professional Services Group
3160 Fairview Park Drive
Falls Church VA 22042

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Rome Laboratory (OCSM)
26 Electronic Pky
Griffiss AFB NY 13441-4514

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-95-120

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Richard R. Gassner/OCSM/(315) 330-3574

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Rome Laboratory (RL) OCSM directorate is actively developing a Multisensor Data Fusion Tracking Testbed capability for evaluating, testing, and developing fusion algorithms. This report describes the work performed by CSC Professional Services Group to improve this testbed, install the University of Connecticut's Multisensor Air Traffic Surveillance (MATSurv) Tracking Algorithm, and validate the algorithm's performance. The MATSurv algorithm is the result of RL's efforts to transition data association and estimation algorithms, developed as part of AFOSR funded research, out of the university environment and into the field for test and evaluation. The testbed is unique in that it provides a menu driven variable configuration capability that allows researchers to configure the software and hardware resources. The resources include data sources, data processing components, and display devices configurable into real world combinations. Thus, the testbed provides a framework for the evaluation and modification of abstract methods of combining multiple sources of data into fused tracks.

14. SUBJECT TERMS

Multisensor multitarget data fusion, Tracking, Surveillance

15. NUMBER OF PAGES

36

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Accesion For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

1.0 Introduction

The Rome Laboratory OCTM directorate is actively developing a Sensor Data Fusion Tracking Testbed (SDFTB) capability for evaluating, testing and developing fusion algorithms. Rome Laboratory is presently developing several types of surveillance fusion algorithms and needs to evaluate these algorithms in a cost effective manner. The objective of this contract is to (1) enhance the capabilities of the SDFTB, (2) install the Multisensor Air Traffic Surveillance Tracking Algorithm "MATSurv" into the testbed and document, and (3) test and evaluate the MATSurv algorithm in the testbed.

The MATSurv algorithm is the result of Rome Lab's efforts to transition data association and estimation algorithms, developed as part of AFOSR funded research, out of the University environment and into the field for test and evaluation. The Rome Lab, Sensor Data Fusion Tracking Testbed (SDFTB), developed by William Berry RL/OCTM will be the test environment for these new algorithms. This testbed is unique in that it provides a menu driven variable configuration capability that allows researchers to configure the software and hardware resources. The resources include: data sources, data processing components and display devices configurable into real world combinations. Thus, the testbed provides a framework for the evaluation and modification of abstract methods of combining multiple sources of data into fused tracks.

This report describes the work performed by CSC Professional Services Group (April 1, 1994 - September 30, 1994) to improve the testbed, install the MATSurv algorithm with the testbed, and validate algorithm performance. In particular the report details the installation of the Multisensor Air Traffic Surveillance, (MATSurv) tracking algorithm as a new tracker type in the Rome Lab/OCTM Sensor Data Fusion Testbed. Sections of this report describe the data flow, mapping and transforms implemented to make the algorithm a variable configuration component.

The report is intended to serve future component installers by providing a description of the testbed's data flow, data structures, data coordinate systems, available transforms and installation process. Finally, this report provides suggestions for improving the testbed's coordinate systems and transformation library.

The installation of the algorithm was performed by Computer Science Corporation's PSG group at Rome Labs. Gerald Bright with the support of Robert Mandry, and Mark Barnell, working under the management of Leonard Gratch completed the effort in October of 1994.

1.1 Approaches enhancing capabilities at Rome Lab

The first part of the effort entailed evaluating the capabilities of the RL/OCTM sensor data fusion testbed. After reviewing the Interface Control Document [1] it became clear that the testbed object descriptions and their object member interfacing functions were well designed for modularity.

An area of concern surfaced concerning on the number of coordinates, two or three, to be used in the testbed. It was decided that the testbed would be made coordinate independent by defining the coordinate system to be part of the tracking algorithm. Therefore, the algorithm itself would include the coordinate transforms necessary to interface data in and out of each algorithm. Additionally, each algorithm would include transforms to interface with the performance display coordinates.

A plan was designed to install the MATSurv algorithm. The object oriented design of the testbed made the design plans easy to implement. The majority of the modifications were made to group inputs and interface data. A new approach which would minimize the combination of transforms required to interface algorithms to the testbed was proposed. The proposal suggested defining a standard testbed coordinate system through which all association, tracker and display objects would communicate. The standard systems suggested were a Geodetic system (latitude, longitude and altitude) or Earth Centered Inertial coordinate system. Both are considered adequate to serve as the standard testbed coordinate system and are already in use in some of the algorithms. Details of each of these system are discussed in Section 3.0.

1.2 Capabilities added to Rome Lab testbed

The enhancement made to the capabilities of RL/OCTM testbed was the extension of the testbed to support complex association and tracking algorithms. The sensor model and input parameters were extended to include sensor measurement noise characterization, sensor altitude, probability of detect and surveillance region volume.

The display function of the testbed was in need of a couple of improvements. The first was to increase speed of reading in and displaying the surface map of upstate New York. The target track display input file was modified to initialize the track map background faster and with greater realism. The second improvement was the capability of the track display function to draw measurements and tracks to the display window. An error was discovered when the measurements were being drawn to the window. The X window routines were being called to draw the measurements were not being used correctly. The parameters passed to the X window routines were changed and the measurements display error was corrected.

It became obvious that an enhanced user interface would be useful, and it was suggested that a graphical user interface (GUI) could be developed using Devguide. The Devguide tool creates an environment where a developer can create a GUI, in which a user can select parameters and options from pull down menus, buttons, scroll bars, and text boxes. A preliminary GUI was developed using Devguide. The software could be implemented in the testbed and coded to be used as an alternative user interface module.

1.3 Installation of MATSurv algorithm

The University of Connecticut's fusion algorithm - MATSurv was received, installed and tested in it's original PC and UNIX based forms. A detailed review was conducted and flow diagrams of the MATSurv code were developed. After review it became clear that the algorithm did not process the input data on a scan by scan basis. The delivered algorithm was given all scan data for a given period of time and ran the association function on the complete set first before processing the associated detection through the track filter. Modifications were made to the data input, association and track modules to interface and process the data on a scan by scan basis.

After review of the RL/OCTM testbed and MATSurv algorithm, several additions were required to install the algorithm into the testbed. The MATSurv algorithm needed a drop tracks capability to be compatible with the RL/OCTM testbed. Additionally, a coordinate conversion from geodetic algorithm track coordinates to testbed display coordinates was needed to complete the installation. The algorithm was modified and tested with the added drop track capability and track to display coordinate transformation. To complete the installation of the algorithm, the inputs and outputs of the MATSurv algorithm needed to be mapped into the available testbed input and output data objects. After installation was complete, the MATSurv algorithm was labeled as the Type 1 Tracker in the testbed configuration file. The installation went better than expected. The

process, procedure, interface code developed are described in detail in Sections 3 and 4 of this report.

1.4 Validation of MATSurv algorithm

Initially, the original MATSurv's performance on the delivered test cases were stored for future reference. After installation of the algorithm was complete, the Type 1 Tracker was exercised on the same test cases. The original and testbed developed track data was compared and the two sets were identical. The installed algorithm's performance matched that of the delivered version.

2.0 Testbed Overview

Rome Lab's OCTM directorate has been active in the area of sensor data fusion since the early 80's. Considerable in-house work was accomplished in the late-1980's. The non-object-oriented, evolutionary development of the testbed software of that time resulted in an overall lack of extendibility. As a result only one track fusion methodology, Multiple Hypothesis Tracking (MHT), was implemented.

The SDFTB is an object-oriented approach to the design and development of a fusion testbed providing a variable configuration capability to the test engineers at Rome Lab. As a result of the object oriented design this testbed structure and software objects have been well documented. To interface the association and track filter components of the MATSurv algorithm we reviewed the overall testbed design. A summary of that design follows. A detailed description of the testbed's design is available in reference (1), which contains a programmers view of the testbed framework.

2.1 Testbed Objects

The testbed's software approach to implementing a re-configurable tracker is started by defining a tracking shell class. This is the beginning of all references to any particular test configuration. All tracker objects belong to this class and are members or sub-members of this single object.. This class includes the objects and sub-objects as shown in Table 2.1.

Table 2.1 Tracker Class Objects and Sub-Objects

	Tracker Shell Class		
Class Objects:	Sensor Suite	Tracker	Display
Sub-Objects:	Sensor (type)	Algorithms (type)	Display (type)
Data-Objects:	Measurement Block	Track State	Meas.& Track History

The tracker class object maintains the configuration information, and provides the data communication ports (functions) to the sensor suite, selected tracker type and display type. The tracker object requests responses from the sensor suite object. The sensor suite object provides functions to get measurements from each sensor in the sensor suite. The tracker object also provides track state data storage functions: allocate, initiate, and free. Additionally the tracker calls the initiate, update, and finish functions to implement the configuration. To make the tracker, type specific, the generic tracker function pointers are made to point to type specific track functions during tracker initiation.

The sensor suite object implements the test engineer's sensor configuration selection by combining sensors from many real or simulated sensor types. As a sensor is selected they are added to the sensor suite sensor object list. Each sensor is a sub-object in itself. The sensor sub-object stores the sensor specific information required by the tracker and display objects. The sensor specific information includes: location, sensor type, measurement type, data source interface, performance model and transformations necessary to communicate as a standardized sensor object.

The display object provides the generalized interface to the particular display type requested. It provides the generic display and data transfer functions used to evoke and fill the data requests made during a display event. The display object receives and displays sensor suite measurement data and tracker class track state data. The data is communicated through the selected display type transformation functions. Display type may vary because of hardware or graphic library availability.

2.2 Installation Roadmap

To support us in performing the installation of the algorithms as a new tracker type Bill Berry gave us a road map (2) to follow during the installation. We were the first to follow the road map and found it very helpful. The road map's major steps are as follows.

1. Define a unique tracker type identifier.

2. Duplicate existing tracker code and option flow to implement a logically new tracker.

3. Replace testbed interface: initiation, update and finish functions of the copied tracker to implement the new tracker type.

4. Build new testbed and tracker configuration files to request the new tracker.

5. Provide data mapping and /or transformations for the new tracker's control variables, measurement input and track state storage and display interface.

These steps represent the path followed during our installation of the 2-D Auction Association with Kalman track filter (type 1) tracker. These steps are covered in more detail in Section 4 of this report.

3.0 Coordinate Systems

This Section documents the testbed's coordinate systems. It also describes the type 1 tracker's expected input, internal and output coordinate systems. Finally, it documents the transformations developed to allow the two to work together.

3.1 Testbed Coordinate Systems

The testbed has four basic coordinate systems. The testbed world coordinate system, a sensor relative coordinate system, track state coordinate system and display coordinate system. The last two are two dimensional systems.

World Coordinate System:

The testbed uses Geodetic coordinates to locate objects on the earth surface. The sensors, tracker and display locations are provided in Geodetic coordinates.

Sensor Coordinate System:

The testbed sensor measurements are stored in the measurement block object pointed to by Vals. These measurements are taken relative to the sensor location. The location is specified in Geodetic coordinates:

Sensor Longitude (Deg.)
Sensor Latitude (Deg.) and,
Sensor Altitude (Ft.).

As shown in Figure 3.1.1 the sensor measures slant range, azimuth, and altitude, if possible for each target in its field of regard. Slant range is the distance from the sensor to the target. Azimuth is the clockwise rotation angle clockwise from north made around the sensors vertical axis. Altitude is derived from the target's own measurements and thus represents the targets height above sea level measured along the target's vertical axis.

Vals[i] is the i'th measurement and Vals[j] the j'th measurement component where:

Vals[i][1] Stores the target measured slant range in Nautical Miles.
Vals[i][2] Stores the azimuth angle in counts. (0 to 4096) = 0 to 360 degs.

The sensor coordinate system did not automatically provide altitude. Target altitude was added to the measurement data set as:

Vals[j][3] Stores measured target altitude in feet above mean sea level when available. The altitude value is set to zero when the value is not available.

Tracker Coordinate System:

The testbed track coordinates are defined relative to a logical tracker origin specified in Geodetic coordinates:

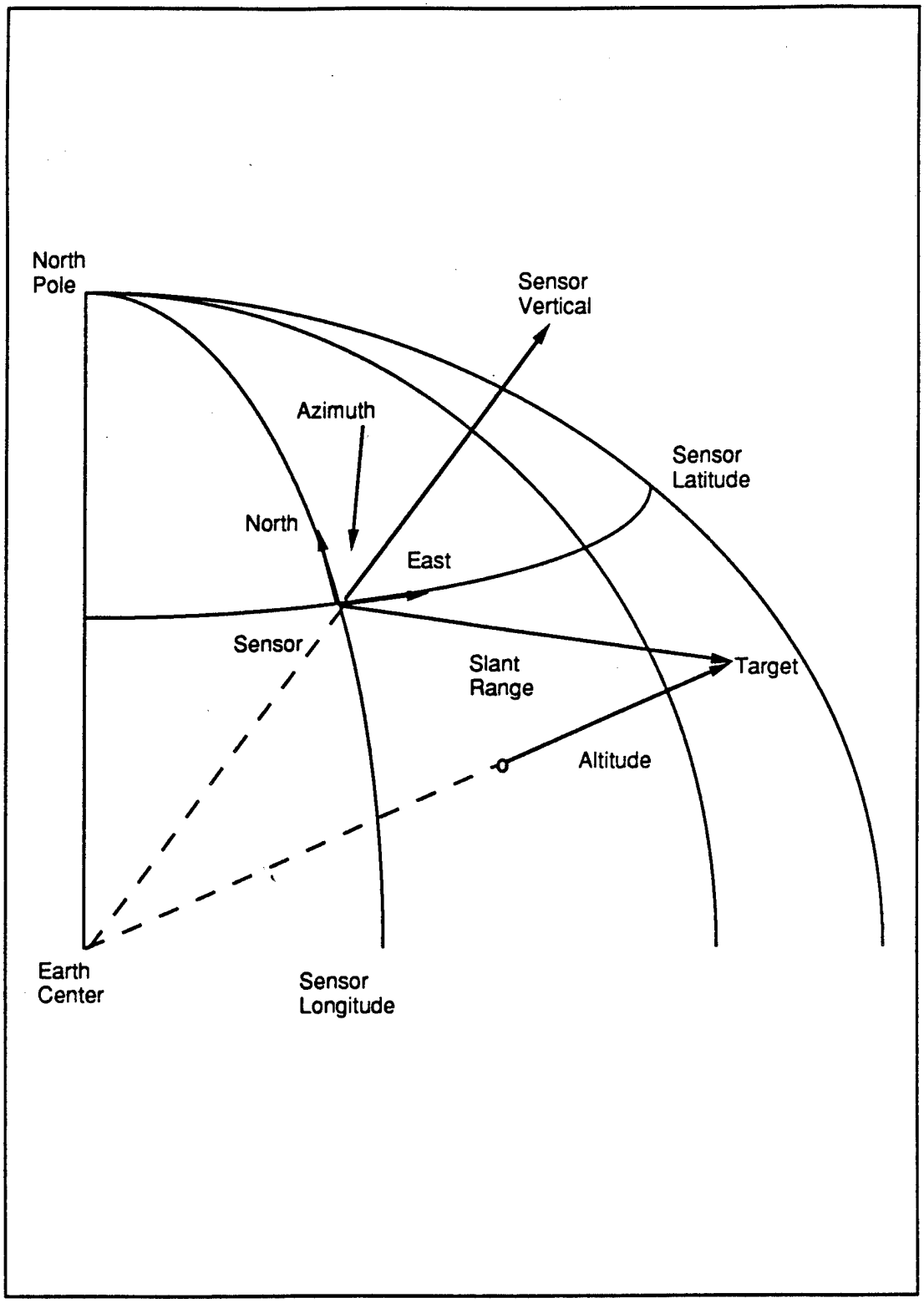

Figure 3.1.1 Sensor Coordinate System

Tracker Longitude (Deg.)
Tracker Latitude (Deg.) and,
Tracker Altitude (Ft.).

The tracker coordinate system is designed to record target tracks locations relative to the logical tracker origin. The track coordinate system axis alignment is: Y = North and X = East taken at the tracker origin as shown in Figure 3.1.2. The tracker's vertical axis extends from the center of the earth through the tracker origin. The North and East axis align a surface normal to a tracker vertical axis. This constant altitude surface is the tracker coordinate "plane". The altitude of this surface is the altitude the tracker - a point on the surface of the earth.

Target tracks maintained in this coordinate system are expressed as distances along the surface in Nmi. The two values maintained in the testbed track coordinate system are:

X[i] = distance east of the logical tracker position. (Nmi).
Y[i] = distance north of the logical tracker position. (Nmi).

Note these distances can be considered arcs lengths taken pivoting at the center of the earth. The arm radius is set equal to the radius of the earth plus the tracker altitude.

Display Coordinate System:

The testbed display coordinates are defined relative to a display upper left origin specified in Geodetic coordinates:

Display Upper Left Longitude (Deg.)
Display Upper Left Latitude (Deg.) and,
Display Upper Left Altitude (Ft.).

The display extent is defined by specifying the lower right Geodetic coordinates:

Display Lower Right Longitude (Deg.)
Display Lower Right Latitude (Deg.) and,
Display Lower Right Altitude (Ft.).

The display coordinate system is designed to display a target measurement or ground track location on a earth surface map of the track area. These maps show the features of the area: cities, roads, airports, lakes, and rivers etc. The display coordinate system axis alignment is: Y = South and X = East taken at the display origin as shown in Figure 3.1.3. The display origin's vertical axis extends from the center of the earth through the display origin. The South and East axis align a surface normal to a tracker vertical axis. A constant altitude surface is the display's coordinate "map plane". The altitude of this surface is the altitude the display origin. Normally this altitude is the altitude of a point on the surface map.

Target measurements and target tracks displayed in this coordinate system are expressed as relative distances along the surface between the upper-left and lower-right coordinates. The two values maintained in the display coordinate system are:

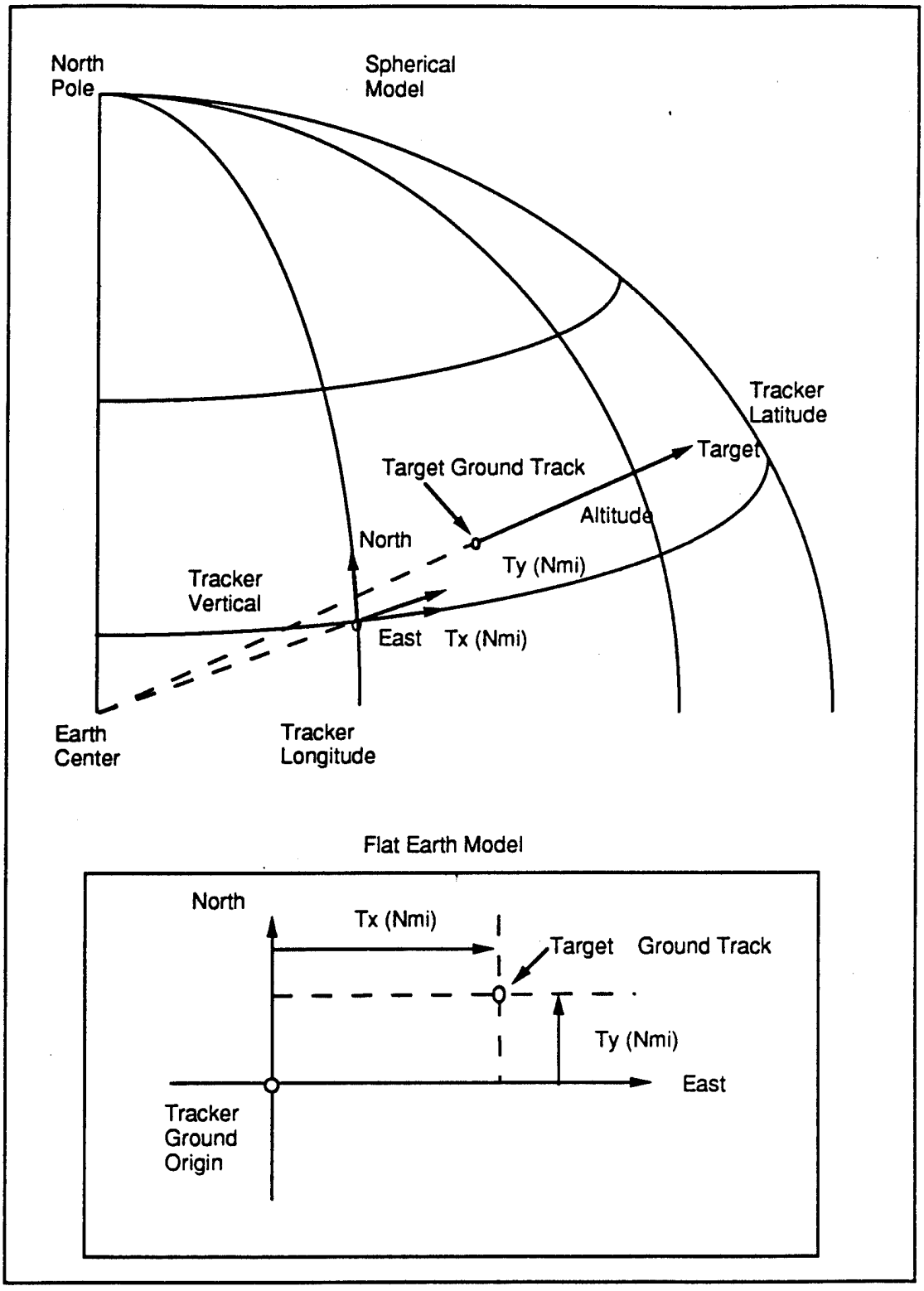

Figure 3.1.2 Tracker Coordinate System

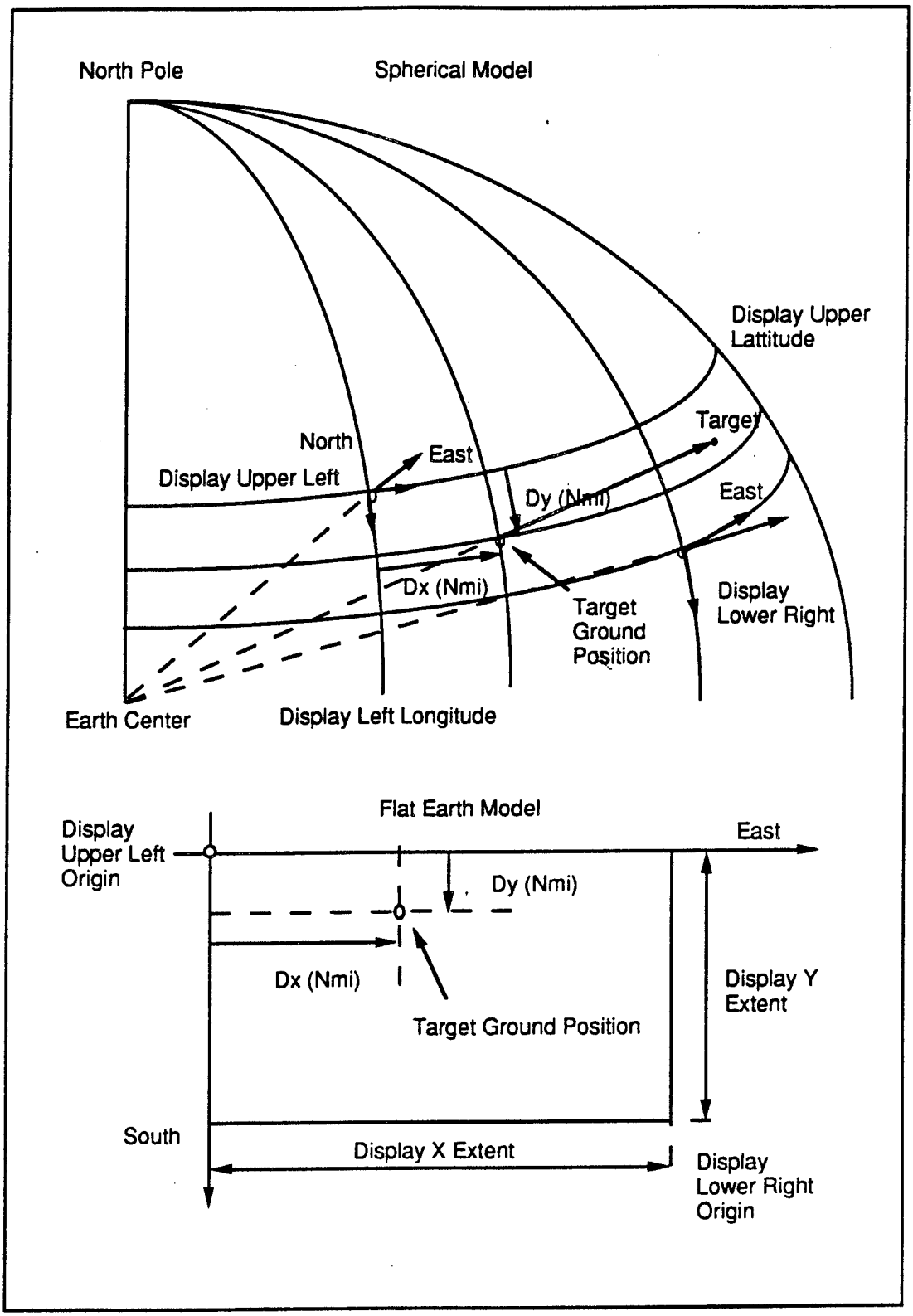

Figure 3.1.3 Display Coordinate System

X[i] = fraction of display width east of the display origin.
Y[i] = fraction of display height south of the display origin.

These offsets are display origin to measurement or track location arc lengths normalized by the display height and width arc lengths. The arc length are calculations are presented in Section 3.3.

3.2 Fusion Algorithm Coordinate Systems

The fusion algorithm came with 4 coordinate systems: Sensor coordinate system, Geodetic coordinate system, Cartesian Linear- Instantaneous target motion system.

The Sensor coordinate system matches the testbed sensor system described in Section 3.1 except that range and altitude units are in kilometers as opposed to nautical miles.

The algorithm converts the "raw" sensor measurements (range, azimuth, altitude) to Geodetic coordinates world (Latitude, Longitude and Altitude) for use by the algorithm. During the conversion from the sensor coordinates to the Geodetic coordinates the target measurements are represented in Earth Centered Rotating (ECR) Cartesian coordinates. ECR means that a Cartesian set of axis are rotating with the earth. The x-y plane is the equatorial plane. The x-axis is aligned with zero degrees longitude. The z-axis extends from the center of the earth out the north pole and the system is a right-handed system as shown in Figure 3.2.1

The Kalman filter "instantaneous motion" coordinate system is another Cartesian system centered at the target track used to calculate the filter innovation values. The testbed never interfaces at this level.

3.3 Data Interfaces and Transformations

To interface the testbed measurements to the algorithm and track results back to the testbed Data transfers, unit conversions and coordinate transforms were developed.

The testbed sensor measurements are copied mapped from the testbed Measurement Block storage structure to scanlist - the association algorithm's measurement storage structure. During this data transfer the range and altitude units are converted from Nmi to Km, and the azimuth is converted from counts to radians.

The algorithm uses these measurement values and the sensor Geodetic location to calculate the target location in Geodetic coordinates. The function raw2geo performs this conversion. The conversion geometry is shown in Figure 3.3.1. The conversion is performed by calculating the sensor location in ECR, converting the sensor slant range vector to an ECR vector , adding the slant range vector to the sensor vector to get a target location in ECR, and then converting the target's ECR location to a Geodetic location. The calculations can be found in the function raw2geo.

At this point the association and track filter algorithms do their stuff and develop target tracks in Geodetic coordinates.

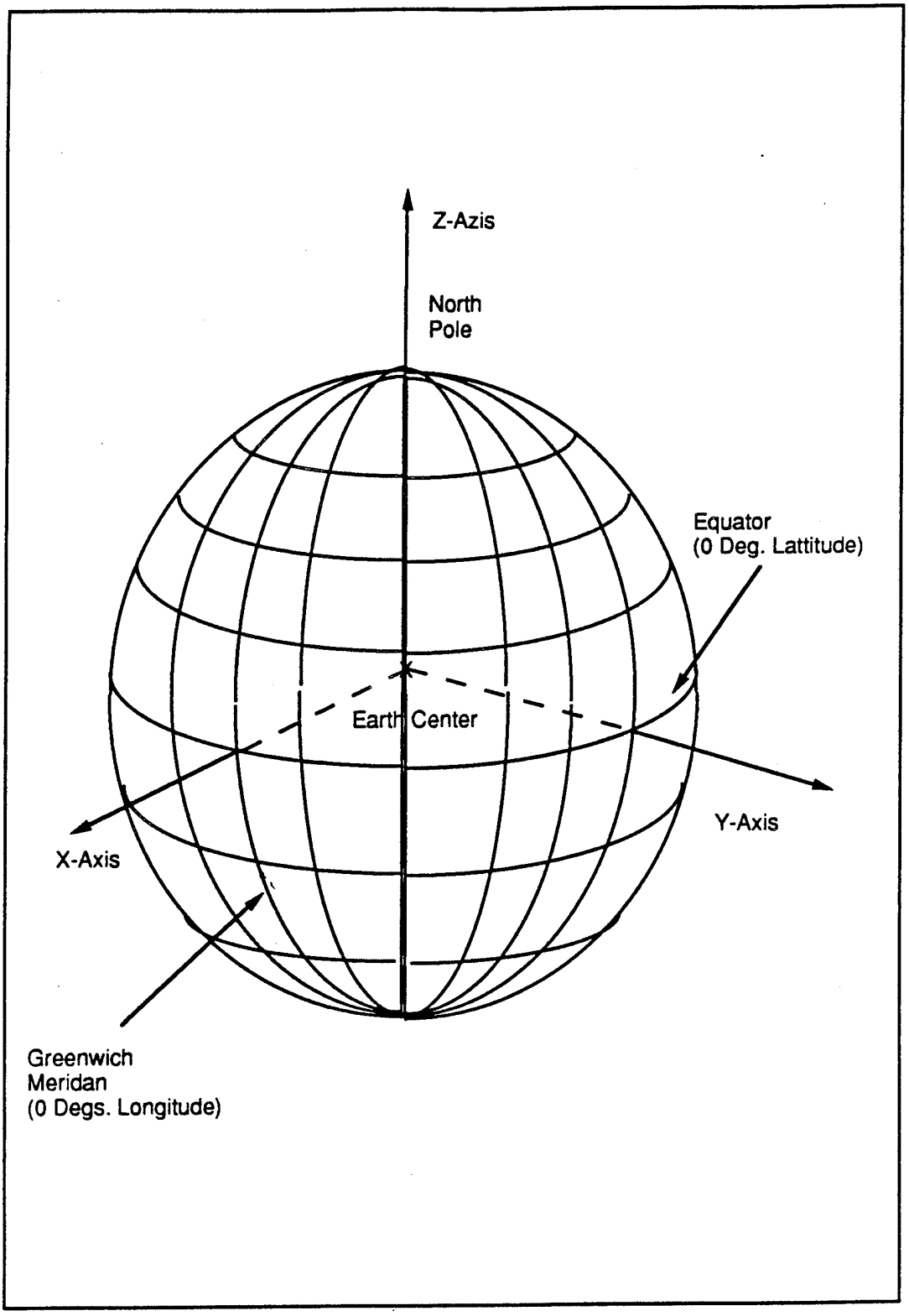

Figure 3.2.1 ECR Coordinate System

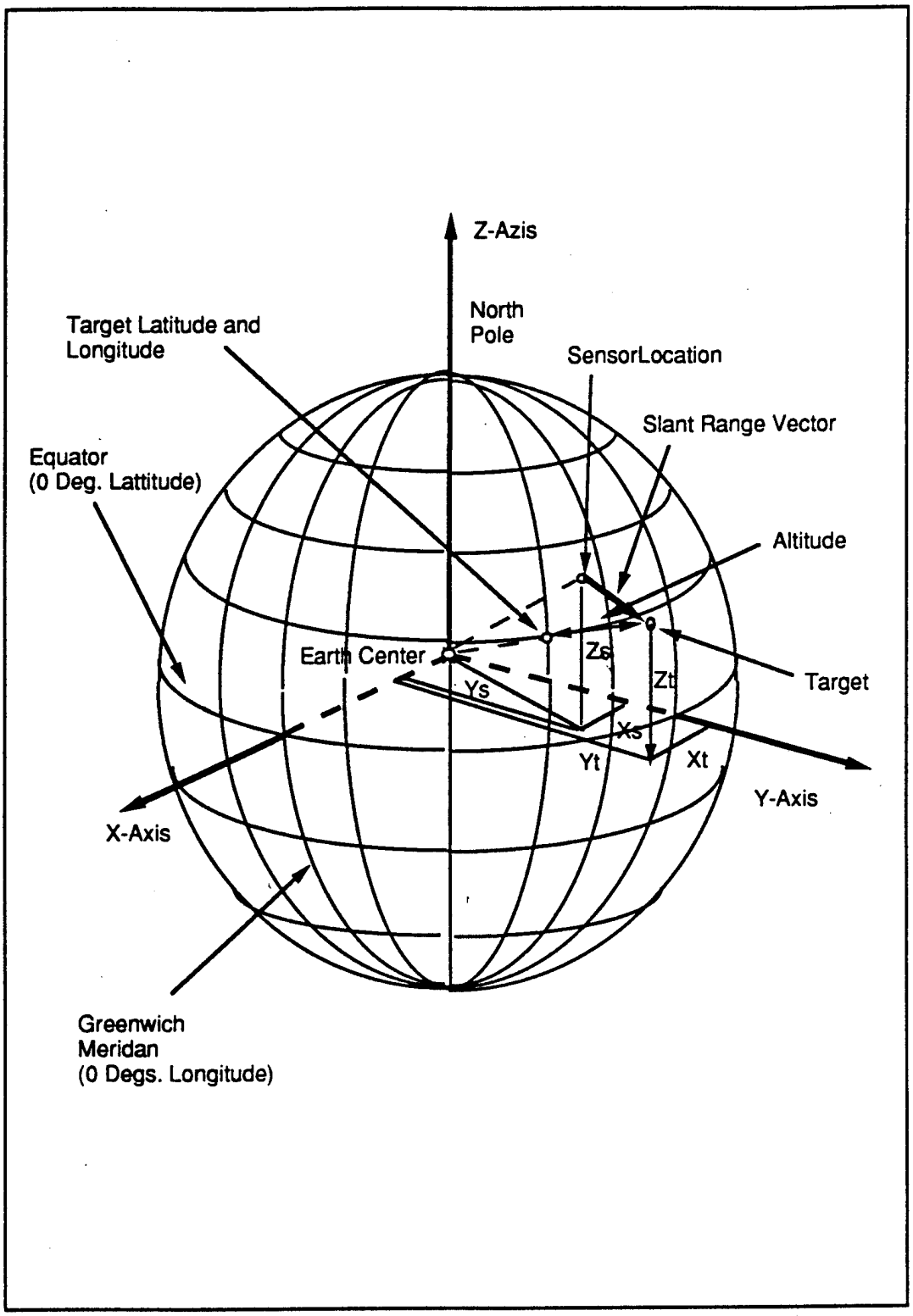

Figure 3.3.1 Sensor Measurement to ECR to Geodetic Conversion

To interface the track results to the testbed track and display coordinate system components we were required to develop new transformations. The goal of these transformations was to get the "ground track" and "ground display" locations.

To get these locations: The first transformation, geo2raw was developed to calculate the target location as if it were sensed by the logical tracker. This results in tracker relative "measured values" of slant range, azimuth and altitude. Replacing the target altitude with the tracker altitude or display altitude we get the "ground" location of the target. Given the Geodetic ground location of the target and logical tracker, the earth centered arc angle between the two points is calculated. This is performed by first converting the two locations to ECR and then using the dot product equation to determine the subtended angle. The arc angle is converted to an arc length by multiplying the arc angle by the total radius, (earth's radius + tracker altitude). The testbed tracking coordinate system, East (X) and North (Y) arc length components are determined by multiplying the separation arc length by the cos and sin of the azimuth angle respectively. The geometry of this transformation is illustrated in Figure 3.3.2.

A similar transformation was developed to get the track results out to the testbed display coordinate system. This transformation uses the display's upper left Geodetic location as a reference and calculates the arc lengths relative to this position but this transformation additionally normalizes the lengths by the display map x sand y arc length extents.

3.4 Future Systems

In this Section we wish to make a suggestion in the area of future development. We suggest developing a standard set of testbed coordinate systems. Presently the user can store any track results in the track storage area. These results then have to be interfaced to all possible display systems. Note that the conversions and coordinate systems discussed above for only one track to display transformation were quite extensive, and the flat earth "arc length" tracker coordinate system is possibly rare.

To minimize work required from future algorithm installers we suggest a review to determine the most popular measurement and track coordinate systems. We should then interface through these systems. Then an installation will require each new component installer to write at most, two transformations necessary to get the data into and/or out of their module. Figure 3.4 illustrates this development environment. The installer will not have to write an interface to all the other modules.

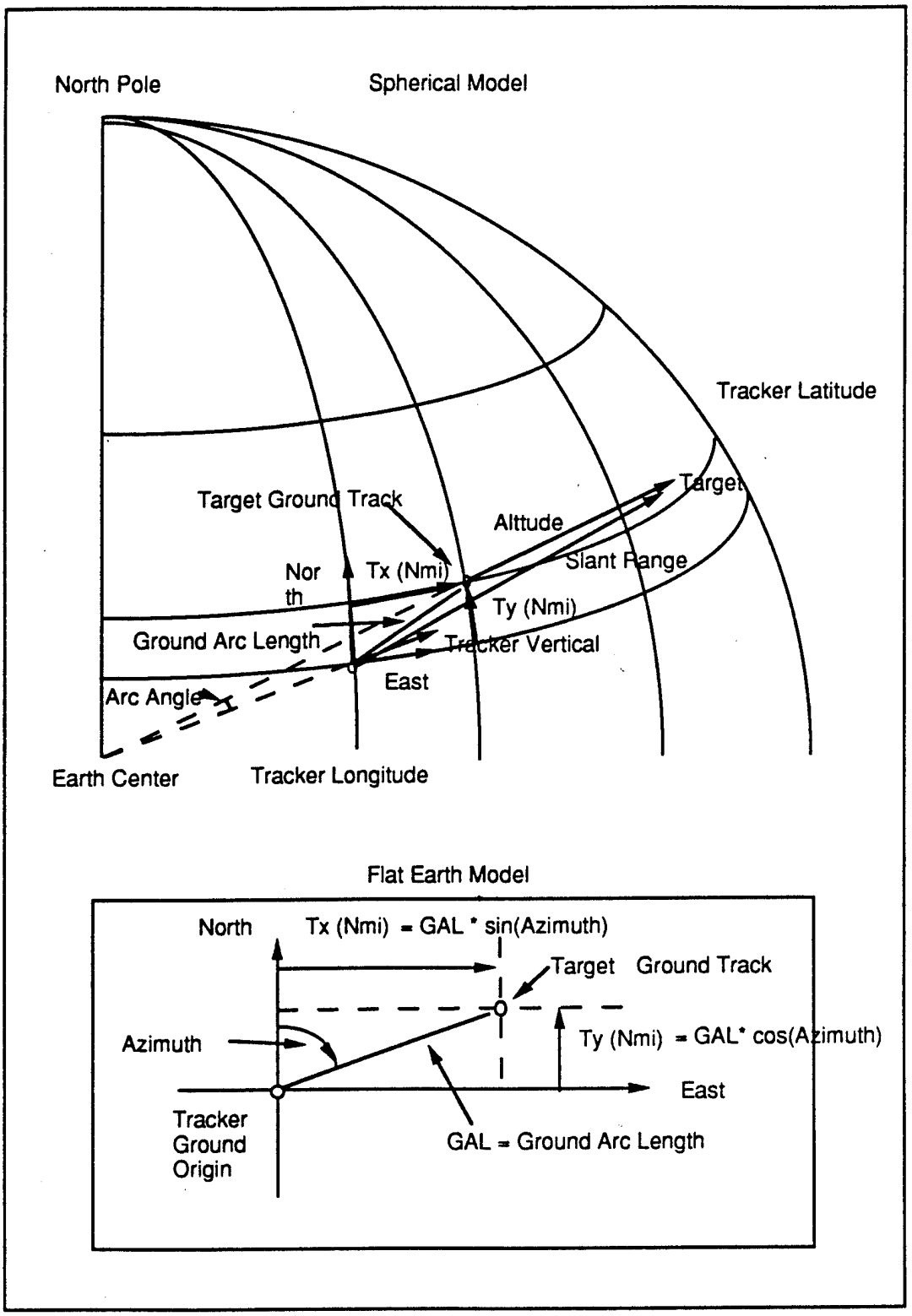

Figure 3.3.2 Kalman Track to Testbed Track Conversion

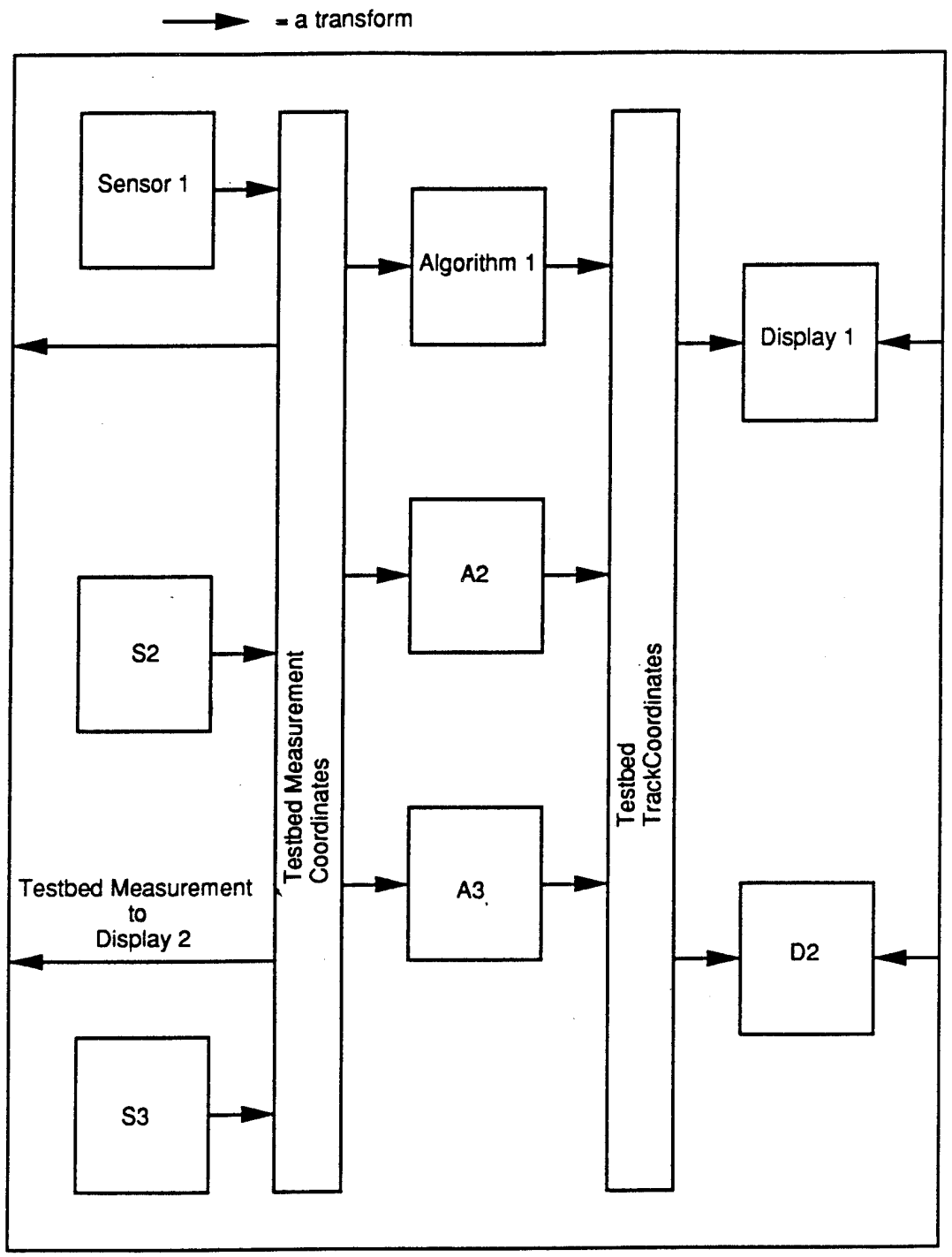

Figure 3.4 Development Environment

4.0 Tracker Installation

In this Section the reader is provided with a software processing flow overview. We present an overview of the testbed flow along with the processing flow of the tracker installed. During the presentation of the flow the new or modified Sections of the code are identified. Finally this Section documents the additional inputs and interfaces coded to install the new algorithm in the testbed.

4.1 Testbed Overview

The RL/OCTM testbed performs three major functions. The three functions are shown in Figure 4.1.1. The first function - Initialization gets the users configuration requests and sets up the logical tracker components. The components are those objects and sub-objects listed in Table 2.1. The second function Get & Process Measurements implements a do until loop that request each sensor of the sensor suite to respond with a measurement set. If a sensor responds its measurements are processed through the selected fusion algorithm. Finally the third function, a Display Event Loop Manager, running in parallel with the second monitors the state of the results and/or state of the display to see if the results or user display request has changed. If so, the functions updates the display as required. Figure 4.1.2 shows the flow, files and functions that initialize the testbed. Figure 4.1.3 shows the flow, files, and functions that implement the Get and Process Measurements function. The display event loop manager was not studied in this effort.

4.2 Tracker Type 1 Development

The process measurements Section of the Get & Process Measurements function implements what ever processing the testbed configuration requests. In this discussion the configuration files have called for an airborne target surveillance tracker. This tracker takes the measured data from each sensor, one at a time, and updates a set of track states. The generic UpdateTracks function called from the process measurements function has been made to point to the new type 1 tracker. The overall configuration and flow are shown in Figure 4.2.1 where the input interface, association, track and output interface are shown.

Input Interface: The input interface is where the measurement conversion from raw to Geodetic takes place. The data in the testbed measurement block MeasBlk, is copied to the algorithm scanlist structure and the sensor coordinates are used to register the measurements into Geodetic locations.

Association: The association of the measurements to tracks is performed over the measurement to track combinations that pass a velocity gate filter. If the apparent velocity of the target using a particular combination does not exceed a limit, the pair is passed on to an association cost estimation function. The cost function for this tracker determines the relative value of using this measurement to update this track. If the cost is reasonable this association possibility is saved along with its cost. Once all reasonable association pairs and their costs have been calculated a near optimal 2-D Auction algorithm finds the best set of pairs that minimize the total cost, assigning only one measurement to each track. A measurement to track association list is returned. If a measurement has not been associated with a track, the measurement list position has a track number of zero, otherwise each measurement list position contains a track number. This list is copied and saved in the testbed association list storage structure.

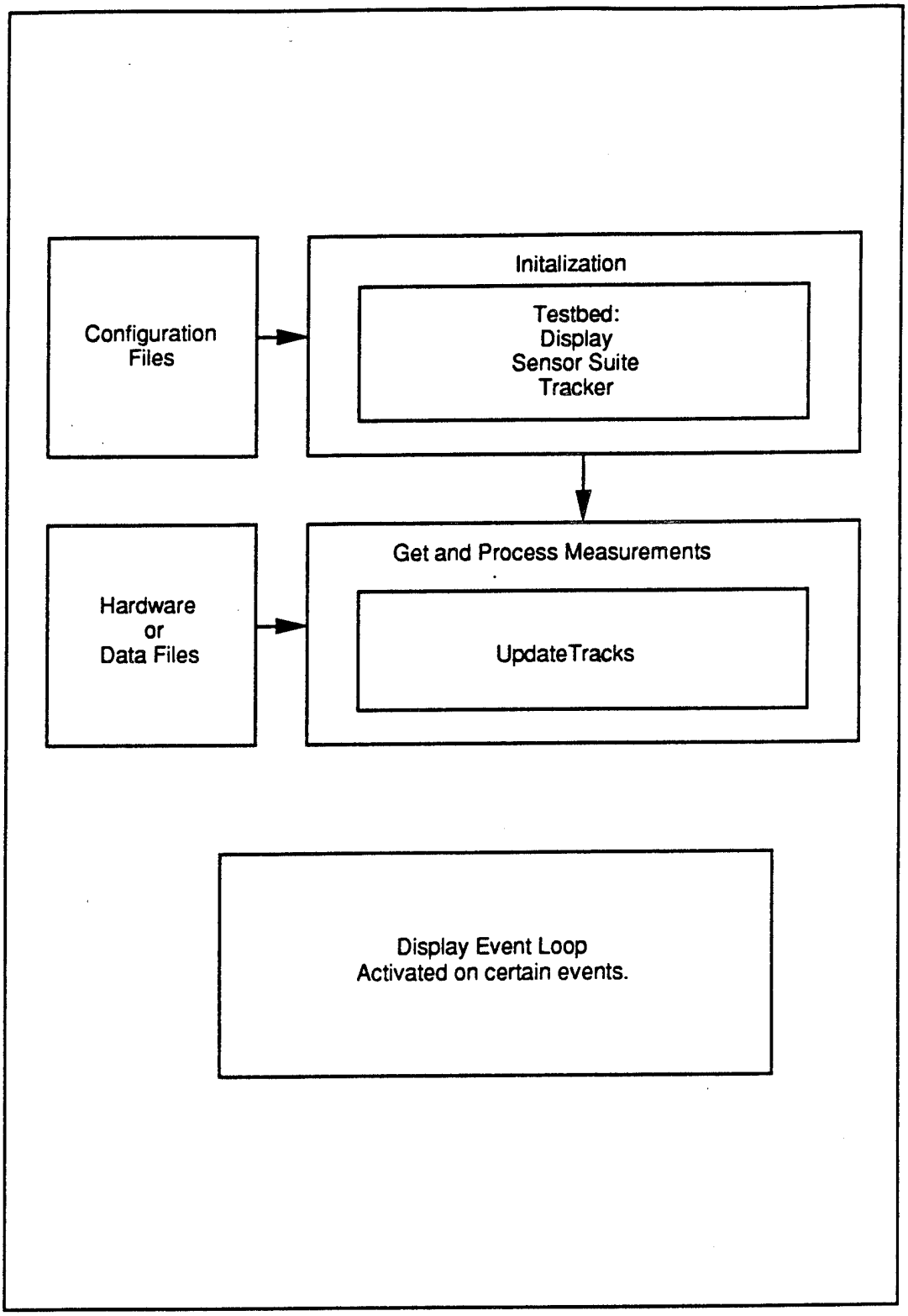

Figure 4.1.1 Overall Testbed Flow

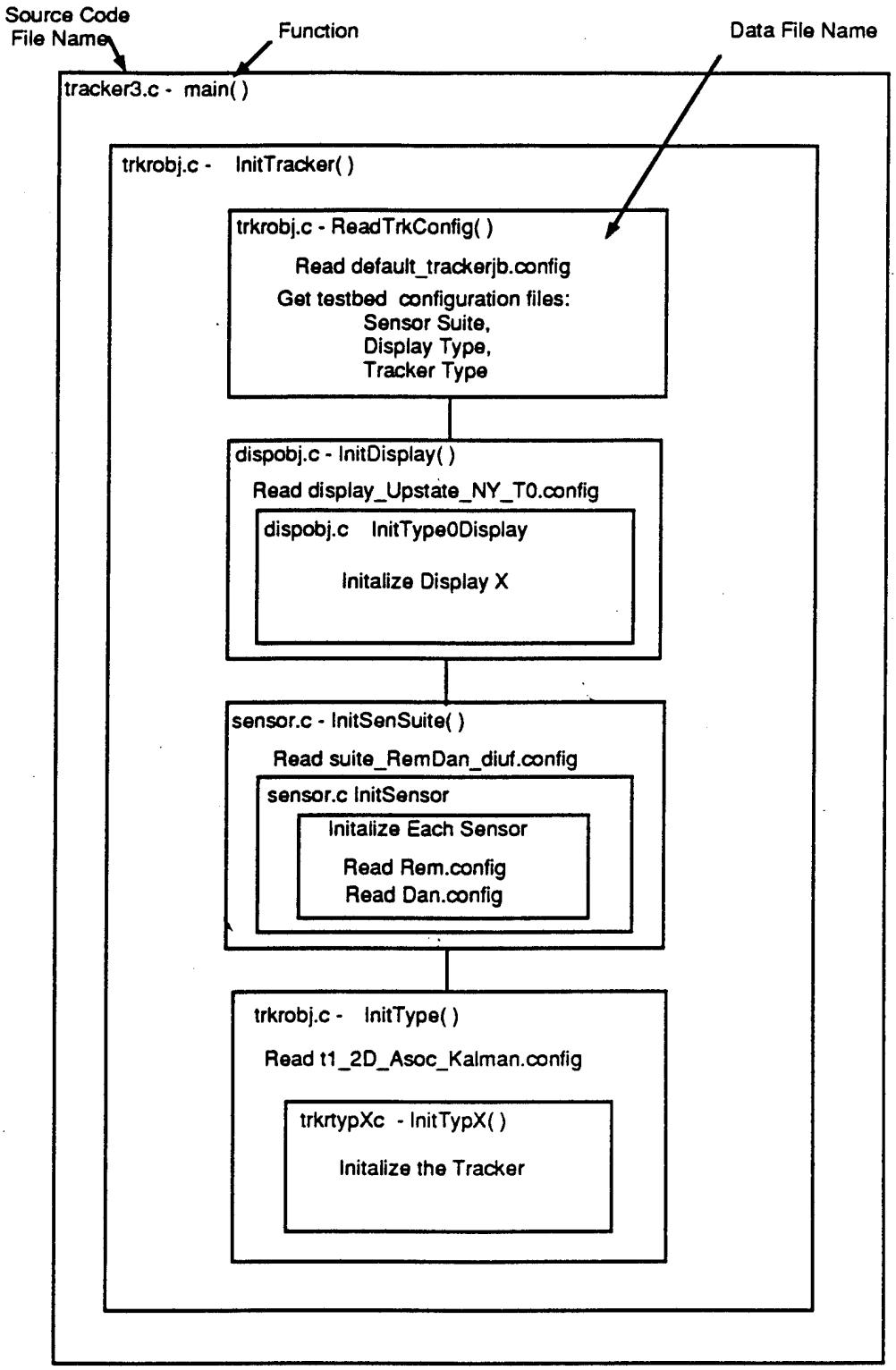

Figure 4.1.2 Flow,Files and Functions that Initalize the Testbed

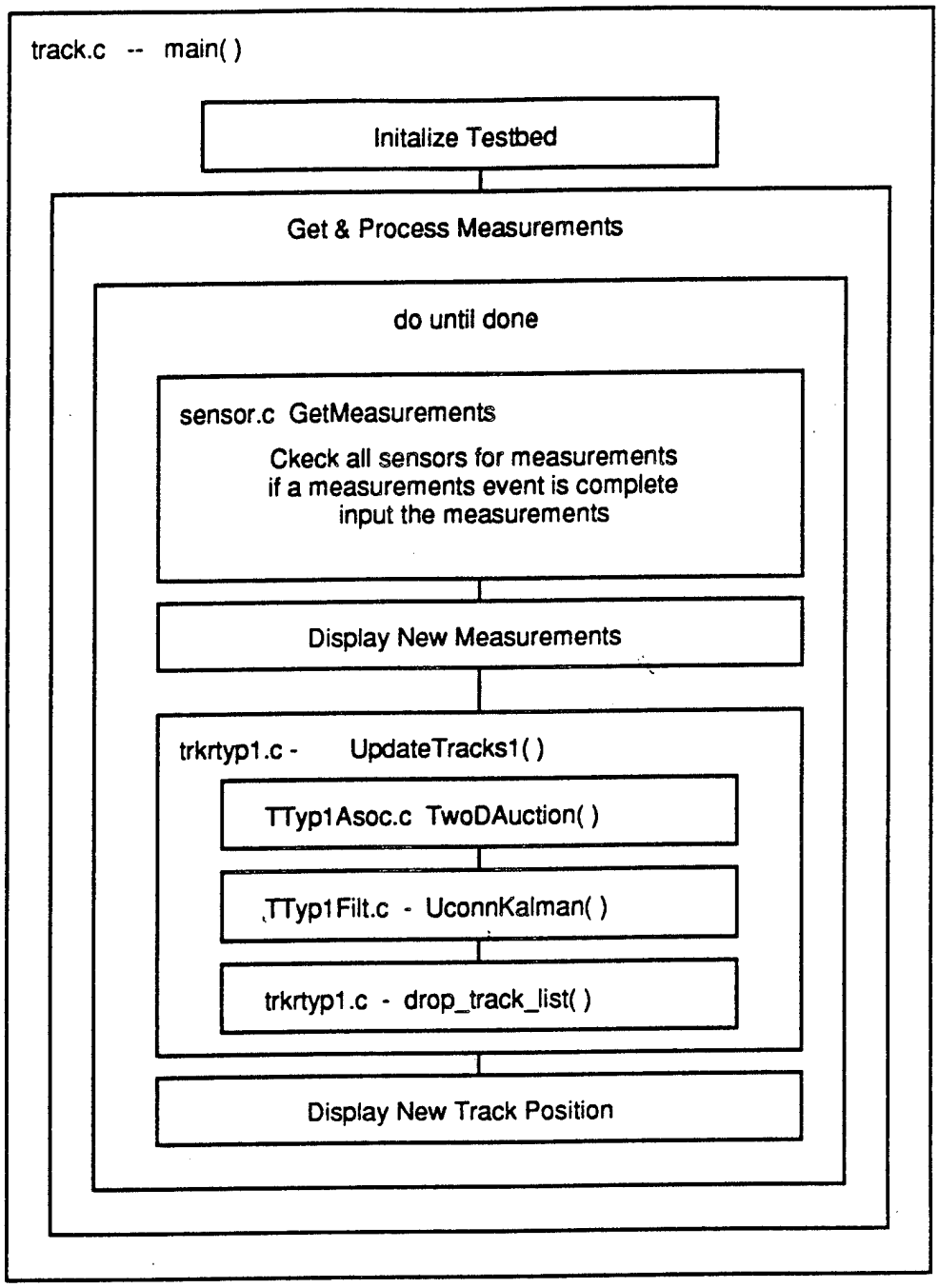

Figure 4.1.3 Get and Process Measurements

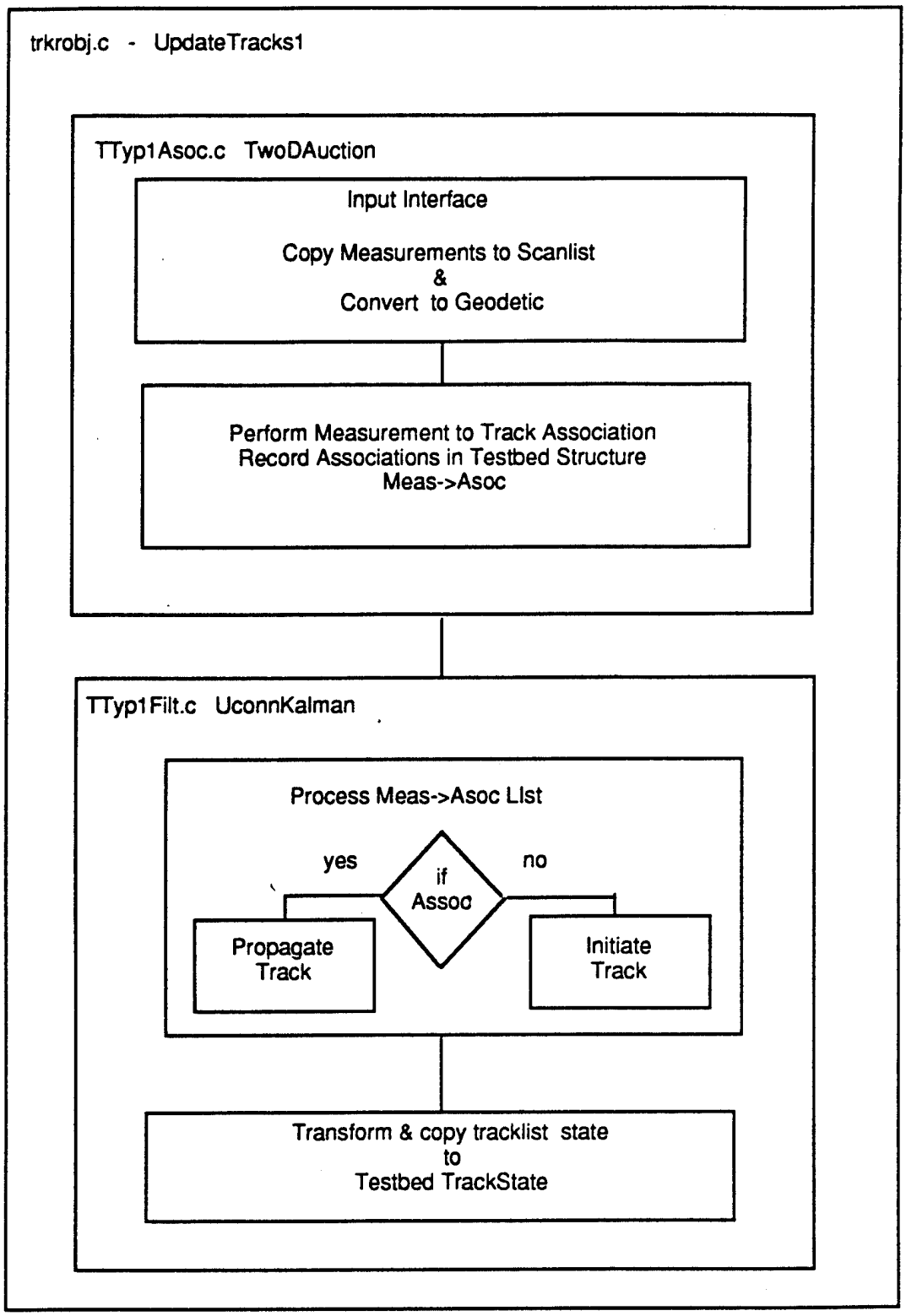

Figure 4.2.1 UpdateTracks1

Track Filter: The track filter process provides the branching between track initiation and track propagation. This process uses the measurement to track association list information to decide to either initiate a new track or propagate an old one. If the list measurement position contains a track number that track is propagated, otherwise a new track is initiated at the unassigned measurement 's coordinates.

Output Interface: This flow chart shows the track list output transformation. The updated Geodetic track states contained in track list are converted to the testbed track coordinates and used to replace or generate new testbed track states.

Now that the flow of the testbed and new tracker have been covered, the details of the installation will make sense. The steps taken to install the tracker are listed below and can be used by new installers to get an idea on how to go about installing a new tracker module.

Installation Steps

1. Define a unique tracker type identifier and copy an existing tracker code renaming all references to the new type number.

2. Copy and reuse the tracker shell type specific processing branches (case statements) to add the new type specific initiation, update and finish options to the testbed.

3. Copy the original tracker type setup file, renaming it to identify the new tracker type. Copy and modify the testbed configuration file to request the new tracker and identify the new tracker setup file.

4. Test the new tracker to see if performs as the original. You now have, as far as the testbed knows, a new tracker type.

5. All trackers are implemented in the testbed by three generic function calls. Initiate, Update, and Finalize. Write replacements for the copied functions to implement your new tracker. To invoke these new initiate, update and drop tracks functions, assign the generic function calls these new function addresses during the branched testbed initiation setup of step 2.

6. Finally, implement the required data mapping.

Type Specific Tracker Controls --

Modify the type specific configuration file to name and contain the required variables. Modify the copy of the original tracker type initialization code to request these variables from the file. Modify the new tracker type object structure to save these control variables. Section 4.3 describes the variables added to implement the type 1 tracker. Additionally, to support the Kalman filter, new sensor information was required. The sensor initiation code was modified to expect these variables and the sensor configuration files modified to include the information. Section 4.4 describes these new inputs.

Input Data ---

Sensor measured data is available when the update function identifies a data sense event has completed for a particular sensor. When the sense event completes, your new

version of the update function should call your algorithm to process the data. The data is available in the reporting sensor object measurement member (Meas).

Output Data ---

Updated testbed track state variables represent the expected output from your algorithm. You can map your internally kept track state variables to the testbed variables or just update testbed state variables as you wish. You can maintain your track data in any state, coordinate system and units you choose. Requests for your type specific track data are passed through a transform supplied by you that provides the track coordinates in units as expected by other objects in the testbed.

Display Data ----

You will be required to supply a function to transform your new track state data to the display coordinates.

These 6 steps represent the path followed during our installation of the type 1 tracker uniquely identified as the 2-D Auction association method and Kalman track filter.

4.3 Tracker Interface

A new testbed configuration file was developed to request the type 1 tracker be used by the testbed. This file was a copy of the "default_tracker.config" file with the association and tracker type requests changed to ask for the 2-D Auction association and Kalman filter tracker. The new testbed configuration file was named type1_tracker.config and is listed in Figure 4.3.1. Additionally a new tracker configuration file was developed from the type 0 configuration file. The type 1 tracker configuration file is named t1_2d_Asoc_Kalman.config. This file as shown in Figure 4.3.1 contains the new tracker control information. In the type 1 tracker case we added inputs to specify Kalman filter process noise values and track and association gate speed and acceleration values. Additionally drop track and negative time sample limits were added. Finally filter processing control variables were added. The display interface was not changed and is listed in Figure 4.3.1 for reference.

4.4 Sensor Interface

The new tracker required information about the sensors that was not available. To make the data available we added the new information at the end of each sensor configuration file. The sensor information added is: Sensor Altitude, Range Measurement Noise, Azimuth Measurement Noise, Altitude Measurement Noise, Probability of Detect and Surveillance Volume. These new files, Rem.config and Dan.config are listed in Figure 4.4.1.

4.5 UpdateTracks1 Function

The UpdateTracks1 function was written by modifying the type 0 UpdateTracks function. The new function calls the type 1 tracker algorithms. This function calls the testbed Associate_Measurement function that has been modified during initiation to point to the Two_D_Auction association algorithm, and then calls the Update function that has been modified during initiation to point to the Uconn_Kalman tracker function. A call to a type 1 drop_tracks function is made after the Update function is finished.

4.5.1 Association Interface

When the 2-D Auction algorithm is called the sensor measurements to scanlist data mapping is performed to interface the testbed sensor data to the type 1 association algorithm. The data mapping copies the measurement data into the appreciate structures of the scanlist structure and converts the units a necessary. This was all that was required to map in the measurement data. The association function generates the Geodetic measurement locations and processes. The function returns a list of measurement to track associations. This list is maintained in the tracker object structure and is available to the testbed.

4.5.2 Track Filter Interface

For every measurement in the input scan the association list gets either a track number or zero indicating not associated. This list is the input to the track filter function. If the list indicates the measurement is not associated with any track a new track is started. If the list indicates a track number the Kalman filter is called to propagate that track with the new measurement. The tracks are maintained in Geodetic coordinates and are available for use by the 2-D Auction association algorithm. It is because of this that the two algorithms can not be separated. They work as a team. The association algorithm solves for the combination of measurement to track associations that would best improve the overall track performance maintained by the Kalman filter by, in a sense, pre-testing most of the track propagates. What is meant is that the association function temporarily propagates a track to see how it would behave. If the track would improve it favors the association and returns a favorable association cost. Otherwise a higher cost.

Once the tracks have been either initiated or propagated the track filter copies the track states out to the testbed's copy of the track states. The track states are converted to the testbed's "ground track" coordinate system. The Geodetic to testbed track conversion is presented in Section 3.

5.0 Conclusions

The introduction of a new tracker type to the testbed has proven that outside personnel can work with and understand the testbed's modular object oriented design. The design and development approach has minimized the work required to add a new tracker module. The coordinate systems required additional documentation and we have provided that documentation in this report. As discussed in Section 3.0's, Future Coordinate Systems, we have suggested a standardized set of coordinate systems to act as interfaces between the modules. If this standardized set is very common, few transformations will be required to interface new modules. It has been a pleasant and enlightening experience to work with such a well designed object oriented testbed.

References

[1] William Berry "Draft Interface Control Document". Rome Lab Technical Document, April 1994.

[2] William Berry, "Road Map to Interface a Set of Tracking Algorithms as a New Tracking Methodology Selectable as a Module in the Fusion Tracker II, Rome Lab Technical Document April 1994.

[3] Mavali Yeddanapudi, Yackof Bar-Shalom, Krishna R. Pattipati, T Kirubarajan, Somnath Deb, "MATSurv -- Multisensor Air Traffic Surveillance" USAF Rome Laboratory Progress Report, March 1994.

Rome Laboratory

Customer Satisfaction Survey

RL-TR-_____

Please complete this survey, and mail to RL/IMPS, 26 Electronic Pky, Griffiss AFB NY 13441-4514. Your assessment and feedback regarding this technical report will allow Rome Laboratory to have a vehicle to continuously improve our methods of research, publication, and customer satisfaction. Your assistance is greatly appreciated.
Thank You

_____

_____

Organization Name:_____(Optional)

Organization POC: _____(Optional)

Address:_____

1. On a scale of 1 to 5 how would you rate the technology developed under this research?

5-Extremely Useful 1-Not Useful/Wasteful

Rating_____

Please use the space below to comment on your rating. Please suggest improvements. Use the back of this sheet if necessary.

2. Do any specific areas of the report stand out as exceptional?

Yes___ No_____

If yes, please identify the area(s), and comment on what aspects make them "stand out."

3. Do any specific areas of the report stand out as inferior?

Yes___ No___

If yes, please identify the area(s), and comment on what aspects make them "stand out."

4. Please utilize the space below to comment on any other aspects of the report. Comments on both technical content and reporting format are desired.

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.